Assignment #C: 五味杂陈

Updated 1148 GMT+8 Dec 10, 2024

2024 fall, Complied by 昂奕,化学与分子工程学院

说明：

1）请把每个题目解题思路（可选），源码Python, 或者C++（已经在Codeforces/Openjudge上AC），截图（包含Accepted），填写到下面作业模版中（推荐使用 typora https://typoraio.cn ，或者用word）。AC 或者没有AC，都请标上每个题目大致花费时间。

2）提交时候先提交pdf文件，再把md或者doc文件上传到右侧“作业评论”。Canvas需要有同学清晰头像、提交文件有pdf、"作业评论"区有上传的md或者doc附件。

3）如果不能在截止前提交作业，请写明原因。

1. 题目

1115. 取石子游戏

dfs, https://www.acwing.com/problem/content/description/1117/

思路：构建一个循环,里面用i记录是谁的轮.一旦a//b>=2 or a==b就跳出循环判断胜负,否则进行一个石子的取

代码：

```
def canwin(a,b):
    return a//b>=2 or a==b
while True:
    a,b=map(int,input().split())
    if (a,b)==(0,0):break
    i=0
    while True:
        i+=1
        a, b = max(a, b), min(a, b)
        if canwin(a,b):
            if i%2==1:
                print('win')
            else:print('lose')
            break
        else:
            a-=b
            continue
```

代码运行截图 （至少包含有"Accepted"）

挑战模式 Python3

```python
def canwin(a,b):
    return a//b>=2 or a==b
while True:
    a,b=map(int,input().split())
    if (a,b)==(0,0):break
    i=0
    while True:
        i+=1
        a, b = max(a, b), min(a, b)
        if canwin(a,b):
            if i%2==1:
                print('win')
            else:print('lose')
            break
        else:
            a-=b
            continue
```

数据有点弱吗？可以申请加强数据

调试代码 提交答案

代码提交状态：Accepted

25570: 洋葱

Matrices, http://cs101.openjudge.cn/practice/25570

思路：一层一层剥.

代码：

```
from collections import deque
n=int(input())
onion=deque([])
for i in range(n):
    onion.append(deque(map(int,input().split())))
ans=0
while len(onion)>1:
    anstmp=sum(onion.popleft())+sum(onion.pop())+sum(onion[i].pop()+onion[i].popleft() for i in range(len(onion)))
    ans=max(ans,anstmp)
if onion:
    ans=max(ans,onion[0][0])
print(ans)
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```
from collections import deque
n=int(input())
onion=deque([])
for i in range(n):
    onion.append(deque(map(int,input().split())))
ans=0
while len(onion)>1:
    anstmp=sum(onion.popleft())+sum(onion.pop())+sum(onion[i].pop()+oni
    ans=max(ans,anstmp)
if onion:
    ans=max(ans,onion[0][0])
print(ans)
```

基本信息

#:	47535809
题目:	25570
提交人:	24n2400011782
内存:	3968kB
时间:	39ms
语言:	Python3
提交时间:	2024-12-03 16:55:30

1526C1. Potions(Easy Version)

greedy, dp, data structures, brute force, *1500, https://codeforces.com/problemset/problem/1526/C1

思路：小顶堆!!!
(因为担心暴力dp会超时所以)
建一个小顶堆,先一直吃药(把药入堆),吃到健康值为负了就把最负的那个药吐出来(heappop).
hard_ver也AC了

代码：

```
import heapq

#处理输入,初始化
n=int(input())
potions=list(map(int,input().split()))
health=0
h=[]#小顶堆

#dp
for p in potions:
    heapq.heappush(h,p)
    health+=p
    if health<0:
        health-=heapq.heappop(h)

#输出
print(len(h))
```

代码运行截图 （至少包含有"Accepted"）

296784850	Dec/16/2024 11:11 UTC+8	_-12-_	C2 - Potions (Hard Version)	Python 3	Accepted	233 ms	25600 KB
296784555	Dec/16/2024 11:04 UTC+8	_-12-_	C1 - Potions (Easy Version)	Python 3	Accepted	77 ms	0 KB

22067: 快速堆猪

辅助栈，http://cs101.openjudge.cn/practice/22067/

思路：
辅助栈是一个单调递减栈,每次放了一只猪,如果比猪堆里其他猪都轻,就在这个栈里记着.
弹出的时候也一起弹出.

代码：

```
import sys
pigs=[]
```

```
dstack=[]
def push(n):
    pigs.append(n)
    if not dstack or pigs[dstack[-1]]>n:
        dstack.append(len(pigs)-1)
def popa():
    if pigs:
        if len(pigs)-1==dstack[-1]:
            dstack.pop()
        pigs.pop()
def min():
    if pigs:
        #print(pigs)
        #print(dstack)
        return pigs[dstack[-1]]

lines=sys.stdin.read().split()
#print(lines)
i=0
while i<len(lines):
    line=lines[i]
    if 'push' in line:
        i+=1
        line=lines[i]
        push(int(line))
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
import sys
pigs=[]
dstack=[]
def push(n):
    pigs.append(n)
    if not dstack or pigs[dstack[-1]]>n:
        dstack.append(len(pigs)-1)
def popa():
    if pigs:
        if len(pigs)-1==dstack[-1]:
            dstack.pop()
        pigs.pop()
def min():
    if pigs:
        #print(pigs)
        #print(dstack)
        return pigs[dstack[-1]]

lines=sys.stdin.read().split()
#print(lines)
i=0
while i<len(lines):
    line=lines[i]
    if 'push' in line:
        i+=1
        line=lines[i]
        push(int(line))
    if 'pop' in line:
        popa()
    if 'min' in line and pigs:
        print(min())
    i+=1
```

基本信息

#:	47542310
题目:	22067
提交人:	24n2400011782
内存:	16724kB
时间:	128ms
语言:	Python3
提交时间:	2024-12-03 22:06:25

20106: 走山路

Dijkstra, http://cs101.openjudge.cn/practice/20106/

思路：(首先需要说明的是...ij在荷兰语中的读音是/ai/,不是狄杰...(话说读成dijiekstra不会感觉哪里怪怪的吗w))

就是Dijkstra,然后没了.具体的思路我写cheatsheet上了,附在作业末尾,写起来还挺多的

代码：

```
import heapq
MAXN=float('inf')

def dij(s,mat,e):#注意,s,e分别是起点和终点.点表示为(w,x,y),w=weight[x][y]

    weight=[[MAXN]*len(mat[0]) for _ in range(len(mat))]
    q=[s]
    weight[s[1]][s[2]]=0
    d=[(-1,0),(1,0),(0,-1),(0,1)]

    while q:
        w,x,y=heapq.heappop(q)
        if (x,y)==e:
            return weight[x][y]
        for dx,dy in d:
            nx,ny=x+dx,y+dy
            if 0<=nx<len(mat) and 0<=ny<len(mat[0]) and mat[nx][ny]!='#':
                new_w=weight[x][y]+abs(int(mat[nx][ny])-int(mat[x][y]))
                if new_w<weight[nx][ny]:
                    weight[nx][ny]=new_w
                    heapq.heappush(q,(new_w,nx,ny))

    return ('NO')

m,n,p=map(int,input().split())
mat=[]
for i in range(m):
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

基本信息

#: 47765336
题目: 20106
提交人: 24n2400011782
内存: 3956kB
时间: 264ms
语言: Python3
提交时间: 2024-12-16 12:52:54

源代码

```
import heapq
MAXN=float('inf')

def dij(s,mat,e):#注意,s,e分别是起点和终点.点表示为(w,x,y),w=weight[x][y]

    weight=[[MAXN]*len(mat[0]) for _ in range(len(mat))]
    v=set()
    q=[s]
    weight[s[1]][s[2]]=0
    d=[(-1,0),(1,0),(0,-1),(0,1)]

    while q:
        w,x,y=heapq.heappop(q)
        v.add((x,y))
        if (x,y)==e:
            return weight[x][y]
        for dx,dy in d:
            nx,ny=x+dx,y+dy
            if 0<=nx<len(mat) and 0<=ny<len(mat[0]) and mat[nx][ny]!='#
                new_w=weight[x][y]+abs(int(mat[nx][ny])-int(mat[x][y]))
                if new_w<weight[nx][ny]:
                    weight[nx][ny]=new_w
                    heapq.heappush(q,(new_w,nx,ny))

    return ('NO')

m,n,p=map(int,input().split())
mat=[]
```

04129: 变换的迷宫

bfs, http://cs101.openjudge.cn/practice/04129/

思路:
Dijkstra,但是有一些问题:

探路探到石头的时候,如果由最短路径走到一块可能未消失的石头上的时间(w+1)通过若干次加二可以得到k的整数倍,就说明可以在石头旁边徘徊来等到它消失然后走上去.
如果可以走上去,就入队并更新相应的时间,如果不行,就不入队
走上去的条件可以证明为not (w%2==0 and k%2==0)
但是如果某个'.'或者四面被石头包围,就不能这样徘徊,得单独考虑.
对于一般的'.',不能入队的条件是k>2 and 四面包围
对于S,不能入队(也就是开场就死)的条件是四面包围
对于E,它即使被包围也得入队,不然就走不到了,入队了也没关系,反正走到它就返回了
代码:

```
import heapq
MAXN=float('inf')

def dead(x0,y0):
    return mat[x0-1][y0]==mat[x0][y0-1]==mat[x0+1][y0]==mat[x0][y0+1]=='#' and mat[x0][y0]=='.' and k>2

def dij(s,mat):
    global k
    weight=[[MAXN]*len(mat[0]) for __ in range(len(mat))]
    q=[s]
    weight[s[1]][s[2]]=0
    d=[(-1,0),(1,0),(0,-1),(0,1)]

    while q:
        w,x,y=heapq.heappop(q)
        #print((w,x,y))
```

```
        if mat[x][y]=='E':
            return w
        for dx,dy in d:
            nx,ny=x+dx,y+dy
            if 1<=nx<len(mat)-1 and 1<=ny<len(mat[0])-1 :
                if mat[nx][ny] in {'.','E','S'} and weight[nx][ny]>w+1 and not dead(nx,ny):
                    weight[nx][ny]=w+1
                    heapq.heappush(q,(w+1,nx,ny))
                elif mat[nx][ny]=='#' and not(w%2==0 and k%2==0) and mat[x][y]!='#':

                    tmp=w+1#找到到这个#的最短路径长
```

代码运行截图 （至少包含有"Accepted"）

2. 学习总结和收获

如果作业题目简单，有否额外练习题目，比如：OJ“计概2024fall每日选做”、CF、LeetCode、洛谷等网站题目。
除了第六题其他题做得都很快...
好久没做计概了,但是感觉今天总体还挺顺的...(不知道是题目简单还是我运气好)
终于弄懂了Dijkstra算法
以及cheatsheet的md版本快弄完了.温故而知新,整理dfs和bfs算法思路清楚了很多
暂时没急着刷题...(罪过)

附:cheatsheet中关于Dijkstra算法的介绍

3-5 Dijkstra算法

- 读作/'daikstrə/,不是dijiekestra...
- 这个算法用于以下情景:一个加权的图(图的每条连线都有一个权重),要求A到B点权重代数和的最小值.
- 例如,几座城市之间修了一些路,每条路有一个长度,要求A城到B城最短的路是多长.
- 对于一般的图,都是如下做法:
  1. 构建邻接表(每个节点跟其他哪些节点连在一起,列一张表).
  2. 构建小顶堆q存储即将遍历的点
  3. 一边遍历一边探路一边更新最小距离(就是bfs)
- 看起来很抽象.直接上代码:

```
import heapq

def dijkstra(n, edges, s, t):
    graph = [[] for _ in range(n)]
    for u, v, w in edges:
        graph[u].append((v, w))
        graph[v].append((u, w))#构建邻接表
    pq = [(0, s)]  # (distance, node)
    visited = set()
    distances = [float('inf')] * n
    distances[s] = 0
    while pq:#遍历节点
        dist, node = heapq.heappop(pq)
        if node == t:return dist
        if node in visited:
            continue
        visited.add(node)
        for neighbor, weight in graph[node]:#走到一个节点-->探路边上的邻居
            if neighbor not in visited:
                new_dist = dist + weight
                if new_dist < distances[neighbor]:
                    new_dist = dist+weight#更新距离
                    if new_dist < distances[neighbor]:
                        distances[neighbor] = new_dist
                        heapq.heappush(pq, (new_dist, neighbor))
    return -1
```

- 这段代码更抽象了,也不够贴近生活实际.

- 在CS101中,我们见到的更多是基于矩阵的Dijkstra算法,因此我们来看看,如果遍历的是矩阵这种特殊的图,Dijkstra算法长什么样.
- Dijkstra的本质是bfs,所以我们拿bfs的代码简单改一下就好了.以下是将一个普通bfs修改为Dijkstra的过程:(注:以下的权重和/权值和就比如说走山路那题的体力消耗量)

    1. 把函数名从bfs改成Dijkstra(确信)
    2. 用小顶堆代替原先的队列q,q中的元素为(w,x,y)元组,其中w是权值和,x,y是坐标.这样每次访问的都是q中权值和最小的点.(实际上这叫优先队列)每次都访问权值和最小的点,使得第一次到终点时的权值和就是全局最小的权值和,也就是第一次到终点的时候就可以直接break而不需要再从其他路径到终点来更新最小权值和.
       注意:heapq的元素如果是元组,会依次按元组中第一,第二...个元素的大小作为该元组的大小比较的依据.所以为了让q按权重和排序,q中的点应表示为 `(w,x,y)` ,其中 `w=weight[x][y]` ,是该点到起点的最小权重和
    3. 建立一个与mat一样大的weight矩阵存储起点到每个点的最小权重和,每个点初值赋为无穷大,除了起点是0(权值和既需要存在这个矩阵里,也要存在q中的元组里,两个地方都需要用到这个权值和)
    4. 开始bfs, `while q` ...这段不变
    5. 探路, `for dx,dy in d` ... 这段,入q条件改成:nx,ny在矩阵范围内,起点到nx,ny的权重和小于 `weight[nx][ny]` (之前的路径走出的起点到nx,ny的权重和)
    6. 在入堆的同时更新 `weight[nx][ny]`

- 代码如下:

```
import heapq

def dijkstra(s,mat,e):#s,e分别是起点和终点.起点s=(0,x0,y0),终点e=(xe,ye).
    MAXN=float('inf')
    weight=[[MAXN]*len(mat[0]) for _ in range(len(mat))]
    q=[s]
    weight[s[1]][s[2]]=0
    d=[(-1,0),(1,0),(0,-1),(0,1)]

    #开始bfs
    while q:
        w,x,y=heapq.heappop(q)

        #先处理到终点的情况
        if (x,y)==e:
            return weight[x][y]

        #然后探路
        for dx,dy in d:
            nx,ny=x+dx,y+dy
            if 0<=nx<len(mat) and 0<=ny<len(mat[0]):#不用not in visited
                new_w=weight[x][y]+______#这段填上点(x,y)到(nx,ny)的权重,如走山路就是abs(int(mat[nx][ny])-int(mat[x][y]))即(x,y)和(nx,ny
                if new_w<weight[nx][ny]:
                    weight[nx][ny]=new_w
                    heapq.heappush(q,(new_w,nx,ny))

    return -1
```

(上面这些能讲明白吗...www)